CPSC 223C

Week 09
Raw Pointers

Lab 15 - Practice Raw Pointers

Complete this lab starting with this side then the back. Write the answers to the questions on your PDF submission.

Fill in the code only using pointer variables

```
#include <stdio.h>

int main()
{
int length; // holds length
int width; // holds width
int area; // holds area
int *lengthPtr; ; // int pointer which will be set to point to length
int *widthPtr;     // int pointer which will be set to point to width
printf("Please input the length of the rectangle\n");
scanf("%d", &length);
printf("Please input the width of the rectangle\n");
scanf("%d", &width);
// Fill in code to make lengthPtr point to length (hold its address)
// Fill in code to make widthPtr point to width (hold its address)

area = // Fill in code to find the area by using only the pointer variables
printf("The area is %d\n", area);

if (// Fill in the condition length > width by using only the pointer variables)
        printf("The length is greater than the width\n");
else if (// Fill in the condition of width > length by using only the pointer variables)
        printf("The width is greater than the length\n");
else
        printf("The width and length are the same\n");

return 0;
}
```

Lab Questions:
Given the following information, fill the blanks with either “an address” or “3.75”.
float * pointer;
float pay = 3.75;
pointer = &pay;

1. printf (“%p”, pointer); will print ___________.
2. printf (“%f”, *pointer); will print ___________.
3. printf (“%p”, &pay); will print ___________.
4. printf (“%f”, pay); will print ___________.
5. Parameters that are passed by ___________ are similar to a pointer variable in that they can contain the address of another variable.

Code	Screen output
`#include <stdio.h>`	
`int main() {`	
`  int vals[] = {4, 7, 11};`	
`  printf(“%p\n”, vals);`	0xfff000bd0
`  printf(“%d\n”, vals[0]);`	4
`  printf(“%d\n”, *vals);`	4
`  printf(“%p\n”, vals + 1);`	0xfff000bd4
`  printf(“%d\n”, *(vals + 1));`	7
`  int* ptr_vals = vals; // take note there is no &`	
`  printf(“%d\n”, ptr_vals[0]);`	4
`  return 0;`	
`}`	

6. **Analyze line #6 and 7. What do you notice about the output of both lines? Why do you think dereferencing *vals* (using *) provides that result?**

7. **Analyze line #5 and 8. Why do you think the addresses differ by 4 bytes? *Hint: What is the data type of each element in thearray?***

8. **Analyze line #10. Why do you think it is valid to assign an array to a pointer?**